VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2019 Tomáš Kořený

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Tomáš Kořený**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: KVADOS, a.s.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Peter Chovanec, Ph.D.**

Konzultant bakalářské práce: Martin Bezecný

Datum zadání: 01.09.2018
Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 23. dubna 2019 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 23. dubna 2019

KVADOS®
SOFTWARE SOLUTIONS
KVADOS, a.s.
Pivovarská 2/1, 702 00 Ostrava 1
IČ: 25826654, DIČ: CZ25826654

Rád bych poděkoval společnosti Kvados a.s. za umožnění absolvování individuální odborné praxe, zvláště pak celému týmu lidí, kteří pracují na produktu VENTUS ®. Mé poděkování patří také Ing. Peter Chovanec, Ph.D. za příkladné vedení práce, věcné připomínky a spolupráci.

Abstrakt

Cílem této bakalářské práce je popsat mou praxi ve společnosti Kvados a.s. na pozici Software Developer. Obsahem mé práce ve společnosti byl vývoj ERP systému zaměřen na řízení skladových zásob zákazníka s implementací neuronových sítí.

Klíčová slova: Individuální odborná praxe, Kvados, Ventus, ERP, SQL, Powerscript, Řízení stavu zásob

Abstract

The aim of this bachelor thesis is to describe my practice in Kvados a.s. as Software Developer. The content of my work in the company was the development of the ERP system focused on customer inventory management with the implementation of neural networks.

Key Words: Individual Professional Practice, Kvados, Ventus, ERP, SQL, Powerscript, Inventory management

Obsah

Seznam použitých zkratek a symbolů

ERP	–	Enterprise Resource Planning
CRM	–	Customer relationship management
SVN	–	Apache Subversion
SQL	–	Structured Query Language
HTML	–	Hyper Text Markup Language
XSLT	–	eXtensible Stylesheet Language Transformations
CSS	–	Cascading Style Sheets
XML	–	eXtensible Markup Language
GUI	–	Graphical User Interface
JSON	–	JavaScript Object Notation

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Odbornou individuální praxi namísto klasické bakalářské práce jsem si zvolil z důvodu propojení praktických znalostí se získanými teoretickými znalostmi během studia a možnost rozšíření si přehledu o svém budoucím povolání.

Do společnosti jsem byl přijat na pozici Software Developer. Mou práci bylo rozšiřovat produkt zvaný VENTUS®. Tento produkt představuje komplexní a ověřený ERP systém pro střední a velké společnosti [12]. Celý produkt je implementovaný v jazyce Powerscript s využitím Microsoft SQL serveru pro správu databáze.

Tato práce popisuje úkoly, na kterých jsem pracoval, dokumentuje mou práci, kterou jsem vykonával a popisuje technologie, které jsem při práci využil.

1.1 O společnosti Kvados a.s.

Společnost byla založena Miroslavem Hamplem v Ostravě roku 1992 pod tehdejším názvem HANAS [5]. Na začátku svého působení firma nabízela software jiných společností. V roce 1993 společnost změnila svůj název na Kvados a o dva roky později se zakladatel rozhodl vytvořit vlastní softwarové řešení a začal se vyvíjet ERP systém VENTUS®, který se roku 1999 nasadil k prvnímu zákazníkovi čímž začal výrazný růst společnosti. O dva roky později se zahájila tvorba dalšího produktu společnosti myAvis®, který se ale vyráběl pod dceřinou společností Kvados mobile solution s.r.o. [5]. Tento produkt slouží jako CRM systém a ve své době byl zcela výjimečný tím, že byl tento systém mobilní [5]. V dalších letech společnost představila další produkty jako například myTeam®, myStock® a myCash®.

V roce 2011 na základě rozhodnutí akcionářů došlo k zjednodušení fungování společnosti a k fúzi KVADOS, a. s., s jeho dceřinou společností KVADOS Mobile Solutions, s. r. o. a dosavadní slogan „Informační technologie“ se změnil na „Software Solutions“ [5].

1.2 Zadané úkoly

Při nástupu do společnosti mi byl přidělen úkol na vytvoření agendy *Předběžné objednávky přijaté*, tento úkol byl vytvořen speciálně pro zaškolení nových zaměstnanců v týmu, zaměřen převážně na pochopení a naučení se práce s využívanými technologiemi a rozvinutí přehledu o produktu a jeho funkčnostech.

Mým hlavním úkolem po zaškolení bylo rozšířit agendu *Řízení stavu zásob* o možnost návrhu množství objednávek dle neuronové sítě. Tato agenda slouží pro pomoc zákazníkům při rozhodování objednávání zboží od dodavatelů. V době mého nástupu již agenda existovala a dokázala doporučit velikost objednávky podle metody klouzavého průměru nebo minimálního a maximálního listingu. Má práce spočívala především v přípravě dat pro neuronovou síť a následnou vizualizaci výsledných dat v přehledných grafech a tabulkách uživateli.

Mým posledním úkolem bylo zajistit novou funkčnost pro hromadné nastavení parametrů výpočtu návrhu množství objednávky. Při práci na tomto úkolu jsem prvně musel odhadnout časovou náročnost splnění a vymyslet postup, který jsem měl následně po diskuzi se zkušenějším kolegou realizovat.

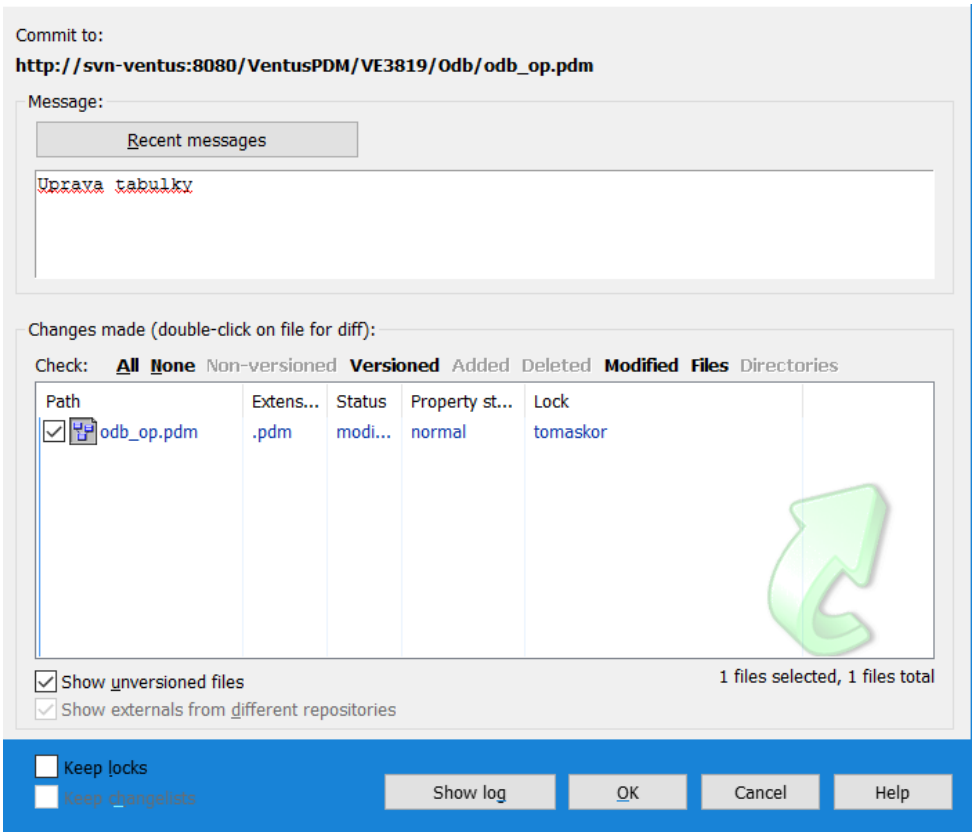
2 Použité technologie

Při práci na prvním úkolu bylo hlavní naučit se používat základní technologie, které společnost používá při vývoji produktu. Ovšem první úkol měl jen naučit základní znalosti a při práci na dalších úkolech bylo potřeba naučit se i jiné technologie, které nejsou ve společnosti používané na denní bázi. Jedná se především o tyto technologie:

2.1 Apache Subversion

Jedná se o open-source systém pro správu verzí [1]. Společnost pro snazší použití využívá klienta TortoiseSVN, který je implementován jako rozšíření prostředí Windows a není vyžadováno používání příkazové řádky [10]. Princip tohoto systému spočívá v ukládání souborů do centrálního uložiště tzv. repositáře [10]. Takže místo ručního psaní příkazů do příkazové řádky jsem mohl použít GUI a nahrát úpravy souborů do repositáře pomocí okna jako na obrázku 1. Pomocí tohoto systému může uživatel sledovat změny provedené uživateli a popřípadě může obnovit starší verzi, pokud by to změny vyžadovaly [10].

Tento systém jsme používali především pro správu databázových modelů a kódu nepřímo souvisejících s produktem jako například šablony pro transformace XML.



Obrázek 1: Použití TortoiseSVN - Nahrání souboru do repositáře

2.2 Powerscript

Powerscript je objektově orientovaný programovací jazyk. Téměř všechny vizuální a neviditelné objekty podporují dědičnost, polymorfismus a zapouzdření [9]. Byl vyvinut v roce 1991 společností Powersoft, kterou roku 1995 odkoupila společnost Sybase. V roce 2010 převzala vývoj jazyka Powerscript a jeho editoru Powerbuilder společnost SAP [6]. Dne 5. července 2016 uzavřela společnost SAP a Appeon smlouvu, na jejímž základě bude společnost Appeon zodpovědná za vývoj, prodej a podporu PowerBuilder [7].

Powerscript tvoří jádro produktu VENTUS® a je zde obsažená velká část aplikační logiky a funkčnosti včetně nadefinovaného vzhledu celé aplikace.

2.3 SQL

SQL je standardní jazyk pro přístup a manipulaci s databázemi [8]. Stal se standardem amerického Národního institutu standardů (ANSI) v roce 1986 a Mezinárodní organizace pro normalizaci (ISO) v roce 1987 [8].

Ve společnosti jsem používal proprietární rozšíření jazyka SQL od společnosti Microsoft a Sybase zvané Transact-SQL [11]. Tento jazyk jsem používal převážně pro psaní databázových procedur, které vytvářeli XML pro generaci webových stránek nebo pro přípravu dat neuronové síti.

2.4 XSLT, HTML, CSS

XSLT je jazyk pro transformaci XML dokumentů vyvinuta konsorciem World Wide Web Consortium [13]. Tento jazyk jsem používal převážně pro transformaci XML, které jsem získal z databázové procedury na HTML, což je standardní značkovací jazyk pro tvorbu webových stránek [3]. Tyto webové stránky sloužili převážně pro snadnější zobrazení informací na menším prostoru, které jsem nemohl zobrazit ve standardních datawindow Powerscriptu. Pro popis způsobu zobrazení elementů webové stránky bylo využito CSS [2].

2.5 Kendo UI

Kendo UI je sbírka komponent JavaScriptu s knihovnami pro jQuery, Angular, React a Vue [4]. Slouží pro budování výkonných webových aplikací [4]. Nabízí komponenty jako například grid, listview, grafy atd [4]. Tuto knihovnu jsem používal pro zobrazení přehledných grafů uživateli na webové stránce s daty o neuronové síti.

3 Agenda Předběžné objednávky přijaté

Mým prvním úkolem bylo vytvoření agendy předběžné objednávky přijaté. Tato agenda měla umožňovat obsluze zadávat předběžnou objednávku přijatou v informačním systému VENTUS®, kde zákazník uvede, jaký sortiment poptává a v jakém množství. Po potvrzení předběžně objednaného zboží ze strany zákazníka, vygeneruje obsluha programu standardní objednávku přijatou z existující předběžné objednávky přijaté. Tato agenda měla sloužit pouze pro studijní účely novým programátorům a nikdy nebyla a nebude použita u zákazníků. K tomuto úkolu byla sepsána analýza, dle které jsem měl vytvořit funkční agendu. Pro tento úkol byla pro mě vytvořena speciální verze systému VENTUS® z důvodu nevytvoření chyb ve vývojové verzi a možnosti více prozkoumat systém bez možných následků ve vývoji.

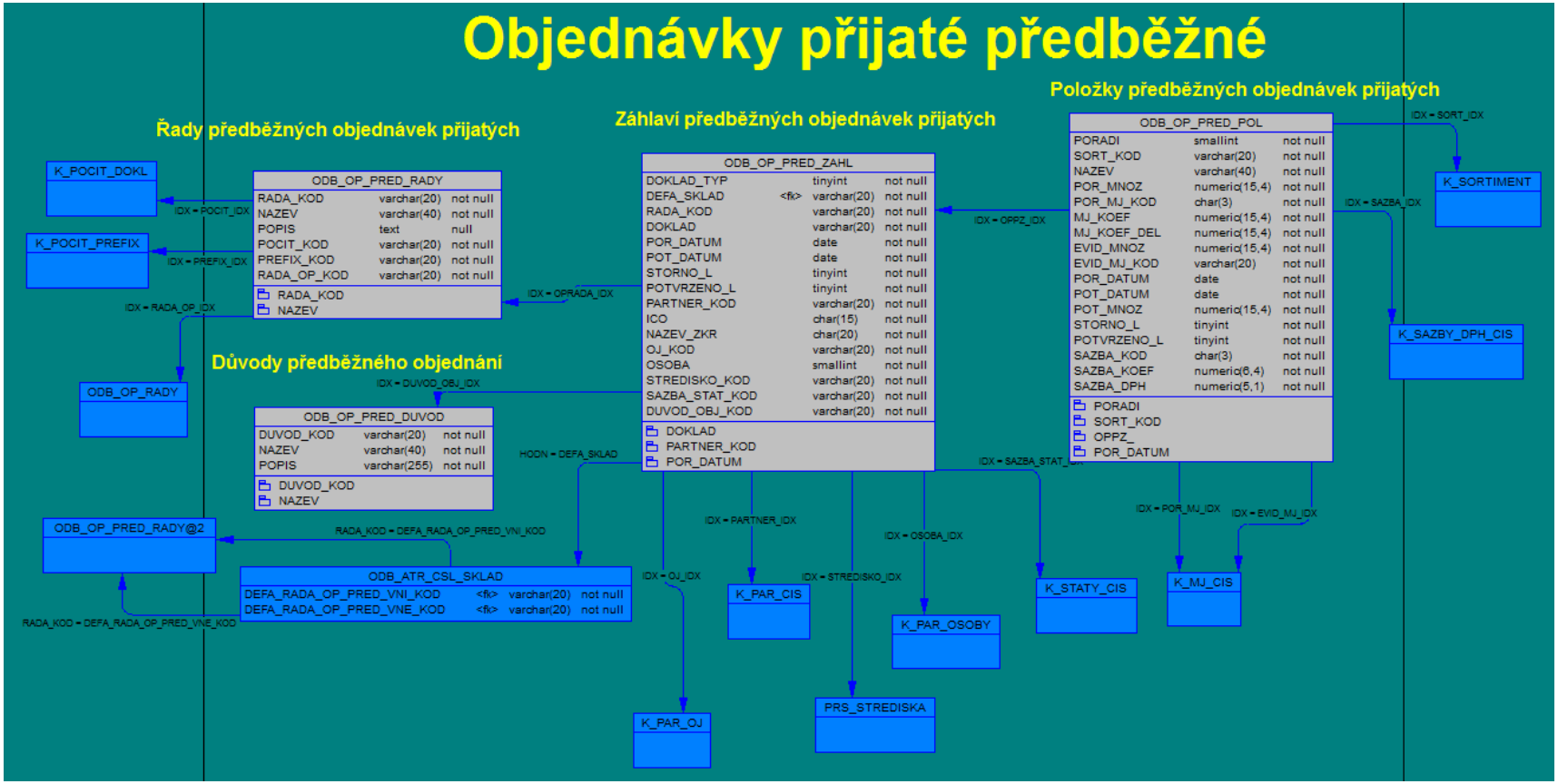
Celý systém VENTUS® je rozdělený do tzv. *agentů*, kde každý agent má svou funkčnost nebo rozvíjí funkčnost jiného agenta. Všichni tito agenti a jejich funkce jsou evidováni v centrálním repositáři, který je stejně jako systém VENTUS® napsaný v Powerscript a dokáže značně zjednodušit práci při rozšiřování produktu. Například dokáže vygenerovat standardní seznamy a detaily pro jednotlivé tabulky, napojení jiných procesů nad seznamy dat tabulek nebo vytvořit jednodušší funkce bez jakéhokoliv programování jako například hromadná editace dat nad seznamem.

První části úkolu bylo tedy se seznámit s centrálním repositářem a vytvořit nového agenta *Předběžné objednávky přijaté*.

3.1 Datový model

Ihned po vytvoření agenta bylo potřeba vytvořit tabulky a registrovat je v centrálním repositáři. Pro vytvoření datových modelů se používá program Data Architect, který usnadňuje práci programátora tím, že nemusí psát SQL příkazy pro vytvoření tabulek ručně, ale stačí je pouze navrhnout v GUI a poté pouze zkonsolidovat do pomocné databáze, z které si centrální repositář přebere všechny důležité informace a tím si také zaregistruje tabulku k agentovi.

V analýze bylo popsáno, jaké tabulky a sloupce je potřeba vytvořit pro veškerou funkčnost dané agendy a stačilo se řídit pokyny. Vytvořil jsem tabulku pro evidenci důvodu předběžné objednávky, řady sloužící pro číslování dokladu a poté tabulku pro záhlaví objednávky a tabulku pro její položky. Tyto tabulky se odkazovali do tabulek již existujících jako například do číselníku sortimentu nebo partnerů. Pokud se přidávají tabulky jiných agentů musí se v modelu nastavit pozadí do modré barvy pro rozlišení, že model neboli agent není vlastníkem tabulky. Výhodou je, že pro tyto tabulky se nemuseli vytvářet jejich sloupce, pouze se mohli přidávat nové, které by tuto tabulku rozšířili. Ovšem pokud jsem přidal do modelu již existující tabulku, tak jsem musel požádat senior programátora, aby v centrálním repositáři nastavil, že můj agent vyžaduje agenta této tabulky. Tímto se zamezí, aby v databázi zákazníků neexistovali tabulky, které by tam být neměly. Datový model měl konečnou podobu stejně jako na obrázku 2.



Obrázek 2: Datový model

Pro zachování standardu sloupců se používají domény, které nastavují základní pravidla a datový typ sloupce. Bývá pravidlem, že každý sloupec má přiřazenou doménu. V některých případech může doména ovlivnit také následné vygenerování datawindow jako například doména pro checkbox, která místo normálního pole pro editaci textu vygeneruje zaškrtávací políčko.

Pro tabulky, sloupce, relace i indexy lze napsat tzv. direktivy, které mohou ovlivnit chování tabulek a jejich sloupců v programu. Tímto můžeme určit přístupnost jednotlivých sloupců závislých na hodnotách v jiných sloupcích, nastavovat hodnoty jiných sloupců, a především zkomfortnit zadávání hodnot polí, které se odkazují do jiných tabulek tím, že zobrazíme podmíněný seznam druhé tabulky a uživatel může přenést hodnotu do stávající tabulky. Například když chceme zadat IČO partnera pro kterého se vytváří předběžná objednávka přijata a neznáme ho. Direktivou určím seznam, které tabulky se má otevřít a obsah kterého sloupce má přenést. Tato direktiva může mít podobu jako výpis 1.

```
#F3TAB=K_PAR_CIS;
#F3RET=partner_kod;
#F3SET1=\GLOB(SUBSTID1=partner_id?=id?);
#F3SET2=ico:V;
#SELECT1=\SUBSTID(partner_id?=0):Wpartner_kod='';
#SELECT2=ico='':Wpartner_kod='':V;
```

Výpis 1: Direktivy nad sloupcem Kód partnera tabulky záhlaví předběžné objednávky přijaté

Po konsolidaci modelu jsem v centrálním repositáři vygeneroval proces pro standardní seznamy pomocí již existující funkce. Seznamy pro řady, důvody a záhlaví předběžných objednávek přijatých jsem zařadil do hlavního menu programu a proces pro zobrazení seznamu položek jsem napojil do menu v okně se záhlavím předběžných objednávek, kde jsem navíc přidal podmínku, aby se zobrazovaly pouze položky označené objednávky. Procesy se na sebe napojují konektory, které jsou již naprogramované a slouží převážně pro předávání dat jednoho procesu druhému.

3.2 Aplikační agent

Po vytvoření modelu následovalo psaní aplikačního agenta. Aplikační agent je speciální druh agenta, který je vždy přidružen ke svému klasickému agentovi a jeho hlavním úkolem je reagovat na události vlastního nebo cizího agenta, kterého rozšiřuje svou funkčností. Stará se o dodatečné operace ve vztahu k databázi jako například integritní kontroly, formátování údajů apod. Jsou jim při databázových operacích nad tabulkami daného agenta zasílány události, na které mohou reagovat. Při splňování této časti úkolu jsem se poprvé seznámil s programovacím jazykem Powerscript.

Úkolem bylo ošetřit mazání již potvrzené položky objednávky, dokladu obsahující alespoň jednu potvrzenou položku nebo dokladu, který už byl potvrzen. Při odstraňování záznamu se

vyvolá událost *before delete* aplikačního agenta, do kterého je poslán chybový objekt, název tabulky, nad kterou byla událost vyvolána a data, která chce uživatel odstranit. Vytvořil jsem funkci, která kontrolovala výše zmíněné hodnoty a v události jsem podmíněně na tabulku položek a dokladů předběžných objednávek přijatých vyvolal svou funkci. Pokud byly splněny některé z podmínek, zaregistroval jsem chybu v chybovém objektu, tím se otevřelo uživateli okno s popsanou chybou a bylo zamezeno odstranění záznamu.

Podobným způsobem jsem musel ošetřit i vkládání nových dat a úpravu již existujících dat. Tentokrát jsem volal své funkce v událostech *before insert* a *before update*. Chyba se zobrazovala uživateli pokaždé, pokud chtěl vložit novou položku do již potvrzené nebo zrušené předběžné objednávky, ale také pokud chtěl upravit již potvrzenou položku.

Poslední části úkolu pro aplikačního agenta bylo předplňovat sklad, z kterého se položka objednává podle výchozího skladu na záhlaví předběžné objednávky přijaté. K tomuto účelu slouží událost *valid* která se dělí na tři typy volání:

- R - událost vyvolána před otevřením okna detailu
- I - událost, která je vyvolána před změnou sloupce a posílá do události *valid* také název měněného sloupce.
- U - událost volána při stisku tlačítka uložit v detailu ještě před samotným zavoláním události *before update*

Má funkce tedy byla vyvolána před otevřením okna detailu, v které jsem si poté načetl vazbu na doklad a zjistil obsah sloupce pro výchozí sklad. Tuto hodnotou jsem poté nastavil také na sloupci sklad položky předběžné objednávky.

3.3 Generace objednávky přijaté z předběžné objednávky přijaté

Jakmile jsem dokončil aplikačního agenta a absolvoval úspěšnou kontrolu mé odvedené práce senior programátorem, mohl jsem se začít věnovat nejdůležitější součásti celého úkolu a sice generaci objednávky přijaté z předběžné objednávky. Prvním krokem bylo založení nového agenta v centrálním repositáři, který bude sloužit jako propojovací agent mezi objednávkou přijatou a předběžnou objednávkou přijatou. Následným krokem bylo vytvoření relačních tabulek mezi položkami a záhlavím. Tyto nové tabulky sloužili pouze pro evidenci a možný náhled z objednávek na předběžné objednávky nebo obráceně. Pro tyto tabulky jsem tedy nevytvářel proces pro zobrazení seznamu, ale vytvořil jsem nový proces pro zobrazení objednávek s podmíněním na existující relaci na objednávku přijatou, které vypadalo jako výpis 2. K takovému podmíněni mi posloužili parametry procesu, do kterých se mohou napsat podmínky pro zobrazení dat. Tyto podmínky se zapisují pomocí direktiv, kde se definují vazby mezi tabulkami, popřípadě použití indexu.

```
#WHEREFROM=1,ODB_OP_OP_PRED_ZAHL_REL,R,FK_OPPZ;###
R.valid=1 AND \GLOB(SUBSTID=R.oppz_id?=%1.id?%) AND \GLOB(SUBSTID=R.opz_id?=
    ODB_OP_ZAHL.id?)
```

Výpis 2: Podmínění procesu seznamu objednávek přijatých

Dalším krokem bylo založení samotné akce v centrálním repositáři a přiřazení konektorů pro napojení této akce nad seznam předběžných objednávek. Bylo potřeba použití dvou konektoru a sice *MENU* pro zobrazení ikony akce v menu seznamu a konektor *CURRENT* pro předání označených záznamu, nad kterými je akce spouštěna. Následně jsem v centrálním repositáři spustil nad agentem regeneraci. Tato regenerace mi do knihovny v Powerbuilderu vytvořila objekt pro obsluhu tohoto agenta, a navíc vytvořila prázdnou funkci generace objednávek přijatých. Navíc se při regeneraci konektor *CURRENT* vytvořil jako instanční proměnná a konektor *MENU* způsobil vytvoření události, která se vyvolá při stisknutí tlačítka v menu.

Zbývalo mi teda už jen napsat kód, který obstará samotnou generaci objednávek. V první části jsem musel ověřit, že je správně napojen konektor *CURRENT* a to z důvodu, že kdyby mou akci někdo v budoucnu chtěl použít nad jiným seznamem a zapomněl připojit tento konektor. Pokud kontrola proběhla v pořádku zobrazil jsem dotaz uživateli, jestli chce opravdu spustit akci pro generaci, pokud nesouhlasil akce byla ukončena. Při odsouhlasení uživatelem jsem si mohl načíst označené doklady s kterými budu dále pracovat. Uživatel mohl označit jeden doklad nebo více, to záleželo, zda chce tuto akci spouštět hromadně nebo jednotlivě. Po načtení všech záznamů jsem v cyklu zkontroloval, zda uživatel nechce generovat ze stornovaných předběžných objednávek. Pokud tak učinil, zapamatoval jsem si tyto doklady a na konci akce jsem jej upozornil, že z těchto dokladů nebylo generováno a při generaci jsem tyto doklady ignoroval. Poté jsem si v další funkci připravil data ke generaci a následně jsem zahájil transakci. Všechno v transakci muselo být v cyklu, aby bylo zajištěno správné ukončení transakce, pokud by nastala chyba. Následně po vložení nového záznamu o objednávce přijaté jsem si musel načíst ID nového záznamu abych vložil záznam do relační tabulky. A nakonec bylo potřeba potvrdit předběžnou objednávku, tím že jsme aktualizovali záznam, z kterého bylo generováno. Tímto jsem vygeneroval objednávku přijatou z předběžné objednávky, ale v transakci bylo potřeba ještě také vygenerovat položky, které měla označená předběžná objednávka. Tady jsme použili stejný postup a opět po vygenerování jsme položku potvrdili. Po vygenerování všech položek jsme teprve mohli ukončit transakci a následně ukončit celou akci s informací uživateli o úspěšném dokončení, popřípadě s informací, z kterých předběžných objednávek nebylo generováno.

3.4 Autoúloha

V poslední části tohoto úkolu jsem se měl seznámit s autoúlohami. Autoúlohy jsou funkce zpracované asynchronně a automaticky automatem z fronty úloh a díky tomuto se uživatel nemusí zdržovat a čekat, než se dokončí složitější operace. Cílem bylo za daných podmínek založit au-

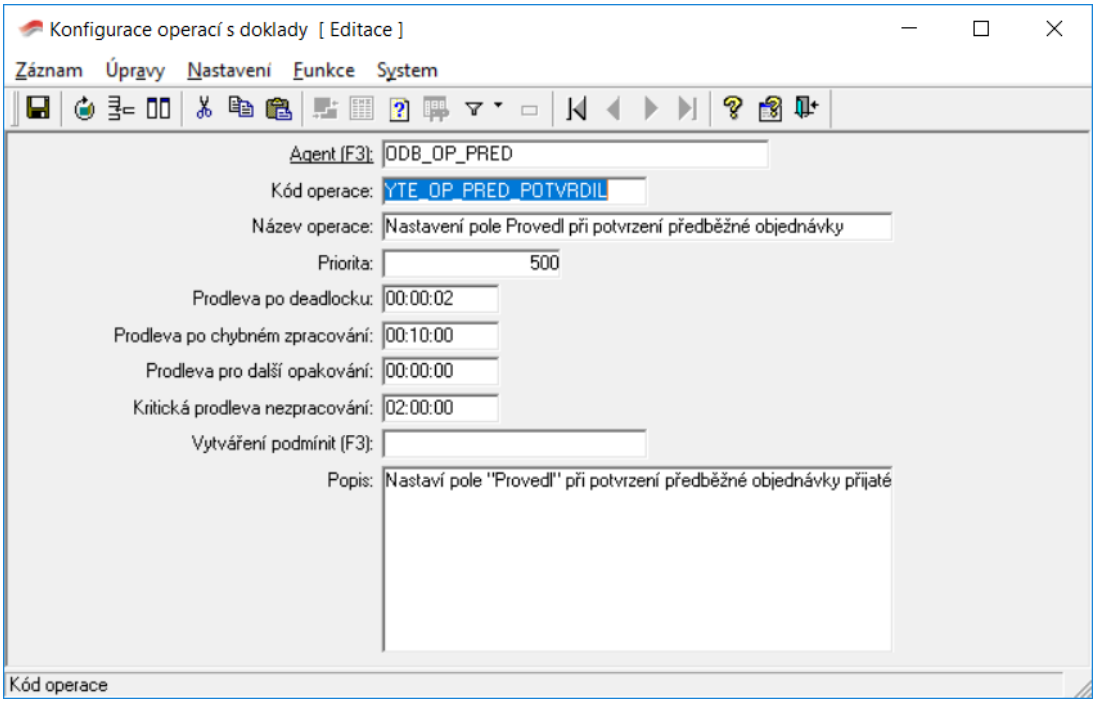
toúlohu, která pouze vyplní pole nad dokladem předběžné objednávky přijaté. Pro takovouto funkčnost by v úkolech pro zákazníka bylo použití autoúlohy zcela zbytečné, ale zde šlo pouze o pochopení, jak se autoúlohy vytváří a jak se zpracovávají. Prvním krokem bylo založení konfigurace autoúlohy v centrálním repositáři. V této konfiguraci se nastavuje název autoúlohy a dále například, kdy se má spustit znovu, pokud při zpracování skončila chybou viz. obrázek 3. Kódy autoúloh se píšou v aplikačním agentovi a pokud je daná autoúloha ve frontě, tak automat vyvolá událost *autoúloha*, kde se podle názvu autoúlohy spustí má funkce.

Po úspěšném založení konfigurace, jsem musel zařadit autoúlohy do fronty zpracovaných dokladů. Toto jsem měl dělat pokaždé, když uživatel ručně potvrdil předběžnou objednávku. Takže jsem v aplikačním agentovi v události *update after* zkontroloval, zda uživatel zaškrtnul pole potvrzeno a pokud ano založil jsem záznam v tabulce fronty zpracovaných dokladů, kde jsem si do parametrů poznačil doklad a jméno uživatele, který objednávku potvrdil.

Při zpracovávání autoúlohy jsem si jako první načetl předané parametry a upravil pole *provedl* dokladu objednávky jménem uživatele, který způsobil založení autoúlohy do fronty. Jako poslední krok mi zbývalo nastavit status autoúlohy, zde byly možnosti:

- 0 - nastav stav autoúlohy na *Zpracovat* a zpracuj autoúlohu znovu
- 1 - nastav stav autoúlohy na *Zpracováno* a odstraň autoúlohu z fronty
- 10 - podobné jako 0, ale neprovede se potvrzení transakce, ale její rollback, takže všechny provedené operace budou navráceny
- 11 - podobné jako 1, ale neprovede se potvrzení transakce, ale její rollback

Protože nebylo potřeba autoúlohu zpracovávat více než jedenkrát nastavil jsem tedy status na jedničku a tím ukončil zpracování autoúlohy.



Obrázek 3: Konfigurace autoúlohy

4 Neuronové sítě

Cílem mého druhého úkolu bylo zpřístupnit moderní predikční matematické metody v ERP systému formou nového způsobu predikce prodeje zboží. Konkrétně mou části bylo připravit data pro neuronovou síť a následně integrovat vypočtená data do informačního systému VENTUS®. Při plnění tohoto úkolu jsem měl využít všechny znalosti získané při plnění prvního úkolu, a navíc využití XSLT šablon pro tvorbu webových stránek.

4.1 Agent a datový model

Stejně jako v předchozím úkolu jsem začal tím, že jsem založil nového agenta, který rozšiřoval základního agenta pro řízení stavu zásob. Následně po založení agenta jsem musel požádat zkušenějšího kolegu o přidělení závislosti nového agenta na základním, aby při instalaci k zákazníkovi bylo ošetřeno, že nový agent nebude instalován bez základního agenta, který zajišťuje základní funkčnost operací.

Při úpravě modelu jsem musel rozšířit tabulku parametrů výpočtu návrhu množství objednávky o sloupec *Interval přípravy dat*, který určuje počet dnů od poslední přípravy množiny dat, než dojde znovu k jejímu rozšíření o nově vzniklá data a o sloupec *Počátek období* určující datum, od kterého budou načteny data pro neuronovou síť.

Tabulka *Parametry výpočtu návrhu množství objednávky* eviduje parametry výpočtu skladových zásob podle způsobu návrhu pro sortiment na určitém skladě. Podle informací obsažených v této tabulce provádíme výpočty, které následně uživateli doporučují, jaké zboží a v jakém množství má objednat na následující období. Pokud v této tabulce existuje záznam vytváří se také záznam v tabulce *Předvýpočty řízení stavu zásob*. Pro záznamy obsahující způsob návrhu *dle neuronové sítě* jsem musel vytvořit navíc vazební tabulku pro množinu dat z neuronových sítí. Pro každý předvýpočet se způsobem návrhu neuronová síť bude existovat záznam v mé nové tabulce *Neuronové sítě – záhlaví* v které budou informace o poslední přípravě dat a ve formátu XML všechna data, která vrátí neuronová síť jako například použité vlivy při výpočtu, počet použitých neuronů sítě, počet iterací atd. Pro každé záhlaví neuronové sítě jsem navíc vytvořil tabulku pro položky, kde byly záznamy pro každý den od počátku období trénování. V tabulce položek jsem vytvořil sloupce pro evidenci data prodeje, což bylo také společně s ID neuronové sítě primárním klíčem tabulky, dále sloupec, ve kterém bylo uvedeno skutečné prodané množství v minulosti, sloupec pro evidenci predikovaného množství, které vrátila natrénovaná síť a sloupec s XML daty ve kterých byly vypočtené vlivy každého dne pro neuronovou síť.

4.2 Autoúloha pro přípravu dat neuronovým sítím

Dalším krokem bylo vytvoření nové autoúlohy pro přípravu dat a výpočet hodnoty vlivů pro každý den. Tato autoúloha se spouští o půlnoci každého dne, aby nebyl vytížen SQL server přes den a tím nebrzdil uživatele při práci. Poběží iteračně a v každé iteraci najde sortiment, pro

který ještě nebyly připraveny data. Pokud již není nalezen žádný sortiment, autoúloha se nastaví, aby se spustila další noc. Pro každý nalezený sortiment zjistíme záznam v předvýpočtech řízení stavu zásob a ID záhlaví jeho neuronové sítě pro následující práci.

Jako první jsem vytvořil dočasnou tabulku, kde jsem si naplnil seznam dostupných vlivů, které jsme si předem nadefinovali. Následně jsem vypočítal hash hodnotu s typem hashování SHA1. Hash hodnotu jsem získal tak, že jsem si tabulku s dostupnými vlivy převedl do XML a to jsem následně pomocí funkce *HASHBYTES()* převedl na hash. Tuto hodnotu jsem porovnal s hodnotou v záhlaví neuronové sítě. Pokud se tyto hodnoty lišili, tak jsem nastavil datum, od kterého se mají připravit data na hodnotu podle parametru výpočtu návrhu množství objednávky. Pokud hodnoty nebyly rozdílné, tak datum zůstávalo stejné podle posledního trénování.

Následovalo získaní hodnot vlivů pro každý den od data posledního trénování. Pro získaní hodnot jsem vytvořil další dočasnou tabulku, která obsahovala záznamy pro každý den do aktuálního. V dočasné tabulce byly sloupce určující sklad a sortiment a dále také obsahovala sloupec pro každý vliv, který byl nadefinován. Následně jsem pro každý záznam v dočasné tabulce aktualizoval hodnoty vlivů. Zjistil jsem například výši standardní ceny sortimentu v každém dni, vlivy související s akcemi jako například počet dnů v akci, cena v akci, sleva v procentech apod. nebo zda byl státní svátek nebo víkend. Jako poslední krok jsem vypočtené hodnoty vlivů převedl do XML a aktualizoval záznamy v položkách neuronových sítích.

4.3 Webová stránka s informacemi o neuronové síti

Když jsem připravil data pro neuronovou síť, tak už nic nebránilo tomu, aby se mohla pro sortimenty na různých skladech natrénovat. Toto umožňoval manažér neuronové sítě, který poté vracel predikované množství prodeje pro následující dny. Tyto data jsem měl za úkol zobrazit v podobě, které by rozuměl také normální uživatel. V agendě řízení stavu zásob již bylo vytvořené okno, které umožňovalo uživateli objednat sortiment a také ve webové stránce zobrazovalo různé informace jako například množství na cestě, rezervované množství nebo také použitý způsob návrhu a parametry výpočtu objednávaného množství. Tyto informace byly rozděleny do záložek a já měl za úkol vytvořit další záložku o informacích neuronové sítě ve které měl být také graf porovnávající skutečný prodej s predikovaným.

Začal jsem tím, že jsem rozšířil databázovou proceduru, která vracela všechny záložky pro webové okno. V této proceduře jsem také nastavil ID okna, abych mohl odlišit jaká záložka se má při rozkliknutí otevřít. Dále jsem rozšířil proceduru, která podle ID záložky, na kterou uživatel kliknul vracela kód procedury, která se má zavolat pro vytvoření XML k transformaci, dále kód XSLT šablony, která bude sloužit pro transformaci na webovou stránku, a také jak dlouho se má soubor držet v cache, než se má vytvořit nový, tak zvané datum expirace. Tento časový údaj jsem tvořil tak, že jsem k aktuálnímu datu přičetl 600 sekund, tento údaj mi pomáhal určovat po jakou časovou dobu bude systém zobrazovat aktuálně vytvořený soubor. Hodnota se zapsala do tabulky, kde společně s časovým údajem bylo zaznamenáno, v kterém adresáři je stránka

uložena. Pokud tedy uživatel chtěl zobrazit stránku, která byla v tabulce evidována a neskončila její doba expirace, tak systém nevytvářel novou stránku, ale zobrazil již vytvořenou.

Následovalo vytvoření samotné procedury, která mi vytvoří XML, kde budou obsažena všechna potřebná data pro novou záložku ve webové stránce. Bylo požadavkem, aby se uživateli zobrazily vlastnosti neuronové sítě, definice parametrů výpočtu, vlivy zohledněné v neuronové síti a pak také graf, zobrazující skutečný prodej pro každý den, množství prodeje, které predikovala neuronová síť a také cenu, jakou produkt měl. Skutečný prodej a predikované množství pro každý den jsem měl v tabulce položek neuronových síti, takže tyto hodnoty převézt do formátu XML nebyl žádný problém. Pro získání standardní ceny a akční prodejní ceny jsem mohl využít již existujících procedur, které pro každý den v nadefinovaném období vracely všechna potřebná data, které jsem si vložil do dočasné tabulky. Z tabulky pro ceny jsem vytvořil XML, které jsem spojil s XML daty prodeje a predikce a tím jsem měl lehce připravená data pro graf.

Pro vytvoření tabulek jsem byl požádán o nadefinování standardu, kde bude dána struktura XML a také šablona pro transformaci, které automaticky vytvoří HTML tabulku. Tento nový standard měl zefektivnit práci při vytváření webových stránek s tabulkami, protože do té doby bylo potřeba každou tabulku transformovat ručně, a to zabíralo zbytečně mnoho času. Kořenový element jsem nazval *CommonTable* stejně jako název šablony pro transformaci. Tento element měl atribut id, který sloužil pro identifikaci při transformaci. Následujícím vnořeným elementem byly *metadata*, určující převážně strukturu tabulky, tyto definice mohly být přetíženy v jednotlivém řádku s daty, pokud by to bylo potřeba. V elementu *metadata* pro každý sloupec byl nadefinován nadpis neboli hlavička tabulky nebo také zda má být hlavička skryta, zarovnání textu v tabulce, datový typ nebo zda se má v patičce tabulky provádět součet hodnot daného sloupce. Dalším elementem na stejné úrovni jako metadata byly už elementy určující řádky. Každý element pro řádek měl atribut určující číslo řádku a v tomto elementu byly pokaždé vnořeny elementy pro sloupce. V elementu pro sloupec jsem mohl přetížit informace v metadatech, ale hlavně zde byly vypsány data pro tabulku. Také zde mohly být nadefinované rozšiřující atributy jako například *help*, který přidával do buňky tabulky otazník, který po najetí kurzoru myši zobrazil námi určený text nebo *annotation* zobrazující malý červený text uvnitř buňky. Tyto rozšiřující atributy už někdy byly použity v jiných tabulkách a já je měl pouze zpřístupnit do standardní šablony. Finální podoba tohoto XML měla podobu přibližně jako je uvedeno ve výpisu 3

Dále jsem musel vytvořit standardní XML pro informace o lazení, které už mělo svou šablonu a já jej pouze upravil pro mé využití. Tyto informace byly vypisovány ve vývojovém a testovacím prostředí a sloužili pro jednodušší odlazení chyb. Z vytvořených XML s daty pro graf, pro vyobrazení tabulek a informacemi pro lazení jsem vytvořil jediné XML, které poté vracela má databázová procedura.

```
<commonTable id="predikce">
    <metadata label="Predikce">
```

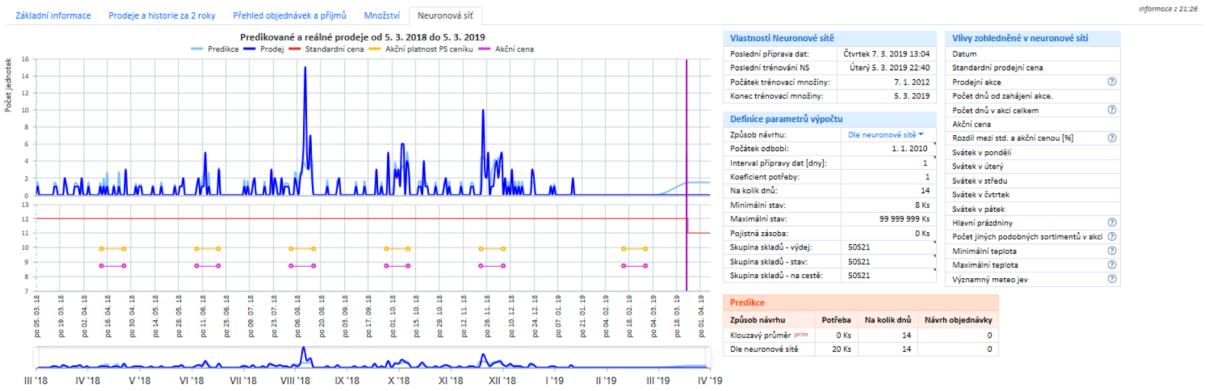
```
      <col number="1" label="Způsob návrhu" type="text" />
      <col number="2" label="Potřeba" type="number" unit="Ks" />
      <col number="3" label="Na kolik dnů" type="number" />
      <col number="4" label="Návrh objednávky" type="number" />
    </metadata>
    <row number="1">
      <col number="1" annotation="prim">Klouzavý průměr</col>
      <col number="2">0.0000</col>
      <col number="3">28</col>
      <col number="4">2.0000</col>
    </row>
    <row number="2">
      ...
    </row>
</commonTable>
```

Výpis 3: Vzor XML pro vytvoření tabulky webové stránky

Pro vytvoření šablony transformace standardní tabulky jsem pouze rozšířil již existující XSLT soubor, ve kterém byly nadefinovány standardní šablony o další šablonu pojmenovanou *CommonTable*. V této šabloně jsem vytvářel tabulku podle údajů uvedených v XML. To znamená, že pokud byla má šablona zavolána pro transformaci vytvořila tag *table* a v něm tag pro hlavičku tabulky, bylo-li potřeba a stejně tak pro patičku tabulky. Mezi vytvářením hlavičky a patičky jsem volal pro každý řádek šablonu, která vytvářela buňky pro tělo tabulky. Tuto jsem vytvářel samostatně hlavně pro lehký přehled a snadnější možnou budoucí úpravu.

Následovalo vytvoření XSLT souboru, který měl sloužit k transformaci vráceného XML z procedury do HTML souboru. V tomto souboru jsem si připojil také XSLT soubor, který obsahoval standardní šablony, které jsem následně pro část XML volal. Pro vytvoření grafu jsem měl využít javascript knihovny zvané Kendo UI, která umožňovala lehkou a rychlou implementaci spojnicového grafu. S touto knihovnou jsem nikdy předtím nepracoval a bylo potřeba nastudovat vše potřebné pro vytvoření požadovaného grafu. Prvním poznatkem bylo, že všechny údaje pro vytvoření grafu se píšou ve formátu JSON a je vkládán do prázdného divu se stejným id, který je vybrán *selectorem* jQuery a nad ním vyvolána funkce *kendoChart*. Začal jsem tím, že jsem si vzal vzorový příklad na stránkách dokumentace pro knihovnu a snažil jsem se jej upravit pro požadovaný vzhled. Jako další krok jsem musel nedefinovat dvě okna (tzv. pane), protože jsem prodeje a predikované množství nemohl zobrazovat na stejné vertikální ose jako prodejní a akční ceny. Poté jsem upravil počet a přejmenoval série. Ty vytvářely svou vlastní čáru v grafu. Takže jsem vytvořil sérii pro prodané množství, predikované množství, prodejní cenu a akční cenu. Jakmile jsem toto dokončil, mohl jsem nadefinovat čtení dat z XML, takže jsem do zdroje dat vytvořil pole javascript objektu, kde byly data pro každý den kromě informací o akčních cenách.

Ty jsem zapisoval až v sérii, protože nebyly určeny pro každý den, ale pouze rozmezím dvou dat a sice začátkem a koncem platnosti akce. Jako poslední krok už mi stačilo jen nadefinovat hodnotu pro horizontální osu, kde jsem pouze určil název z pole zdroje dat čímž byly hodnoty *day*. Tím jsem vytvořil graf a potřebné tabulky a předal kolegovi k vytvoření CSS souboru se vzhledem, aby stránka byla uživatelsky přívětivá. Po úpravě vzhledu mohla záložka vypadat jako na obrázku 4.



Obrázek 4: Vzhled záložky neuronové sítě

5 Hromadné nastavení parametrů řízení stavu zásob

Mým posledním úkolem v rámci mé stáže ve firmě jsem měl vytvořit novou funkčnost v agendě řízení stavu zásob. Nová funkčnost měla podle definic, které uživatel vytvoří v tabulce založit nové nebo aktualizovat již stávající záznamy v tabulce parametrů výpočtu návrhu množství objednávky podle zadaných ostatních parametrů. V definici měl uživatel zadat sklad či skupinu skladů, partnera nebo skupiny partnerů a parametry které chtěl nastavit. Funkce měla být spustitelná z příkazové řádky nebo také ze seznamu definic, kde uživatel mohl vybrat záznamy pro které chce funkci spustit. Jako první krok pro splnění tohoto úkolu jsem měl stanovit časovou náročnost a zároveň vymyslet řešení popsané funkčnosti v zadání. Byla to pro mě první kalkulace a musel jsem překonat velký pocit nervozity, abych neudělal chybu. Naštěstí jsem se vždy, když jsem byl v nesnázích mohl jít zeptat kolegů z týmu, kteří byli ochotni mi vždy pomoct.

5.1 Tabulka definice pro hromadné nastavení parametrů

Základem nové funkce byla nová tabulka, která byla svými sloupci velmi podobná tabulce pro parametry výpočtu návrhu množství objednávky s tím rozdílem, že místo sortimentu zde byly sloupce pro partnera a skupiny partneru, a navíc ještě byla možnost zadat skupinu skladů. Navíc zde existoval sloupec, zda se jedná o aktivní definici, pro kterou jde nová funkčnost zavolat a také sloupec pořadí, ve kterém byla hodnota automaticky doplňována hodnotou posledního pořadí definice, ke které byla přičtena jednička.

Další zvláštnosti bylo, že jsem pro každý sloupec s parametrem vytvořil navíc zaškrtávací pole, které určovalo, zda je hodnota zadaná. Toto řešení jsem zvolil po mnoha pokusech a diskuzích, jak vyřešit můj problém s tím, že VENTUS® standardně nepodporuje vkládání NULL hodnot do databáze. Tato hodnota byla pro mě potřebná pro budoucí rozeznání, zda uživatel zadal určitou hodnotu nebo chce hodnotu parametru převzít z nadřazené skupiny nebo nechat hodnotu z již existujícího záznamu. Dále bylo potřeba napsat direktivy, které po dvojkliku myši do pole partnera, skupiny partnerů, skladu nebo skupiny skladů otevřeli seznam pro vybrání hodnoty a také direktivy, které zakazovali nebo povolovali editaci, protože mohl být vždy zadán pouze sklad nebo skupina skladů a zároveň partner nebo skupina partnerů, nikdy obojí. Zaškrtávací pole byla ve výchozím stavu zaškrtnuta, měla být přístupné až po dvojkliku myši a zároveň při zadání hodnoty do příslušného sloupce se pole mělo odškrtnout, k tomuto mi sloužily další direktivy a tím jsem se vyhnul psaním jakékoliv logiky v kódech a úprav definic datawindow.

5.2 Kontrola na nové záznamy

Aby bylo zajištěno, že spuštění funkce proběhne v pořádku a zároveň, aby nevznikali problémy s daty bylo potřeba vytvořit kontrolu na vkládání a aktualizaci záznamů definic. Nesměla vzniknout aktivní definice, pokud již existovala jiná aktivní definice se stejným partnerem a skladem. Tato kontrola musela hlídat hlavně situace, kdy zadaná skupina partnerů nesměla obsahovat

partnera se zadaným skladem pro kterého již existovala jiná definice se stejným skladem. Taková situace mohla nastat například také, pokud byl partner ve více skupinách najednou.

Tuto kontrolu jsem volal před uložením záznamů do databáze. Načetl jsem si zadanou skupinu partnerů, partnera, skupinu skladů, sklad a pořadí a vyvolal jsem databázovou proceduru ve které jsem si vytvořil čtyři dočasné tabulky pro zadané hodnoty kromě pořadí. Pokud uživatel zadal skupinu, vždy jsem si zjistil partnery nebo sklady které do dané skupiny patřily a poté jsem si zpětně zjistil do jakých skupin patřily zjištění partneři a sklady pro případ, kdy by partner nebo sklad byl ve více skupinách. Pokud uživatel zadal pouze sklad nebo partnera, tak jsem do dočasné tabulky uložil jeho hodnotu a zjistil jeho skupiny. Jako poslední jsem provedl dotaz, kde jsem hledal záznamy definice, které měly hodnoty obsažené v dočasných tabulkách a zároveň měly jiné pořadí než záznam, pro který byla provedena kontrola. Procedura mi vracela pořadí jiné definice se stejnými nebo podobnými hodnotami, pokud existovali. Pokud procedura vrátila nějaké pořadí, vypsal jsem uživateli informaci o existující definici a nepovolil jsem uložit záznam.

5.3 Vytvoření funkce

Když jsem měl za sebou tvorbu tabulek s kontrolami mohl jsem pokračovat s vytvářením funkce. Vytvořil jsem akci v centrálním repositáři a připojil ji na proces standardního seznamu definic. Následovalo samotné psaní kódu funkce. Jak již bylo zmíněno, akci mohl uživatel spustit nad označených záznamech v seznamu definic nebo z příkazové řádky s nepovinným parametrem pořadí, ve kterém byly čárkou oddělené číslice určující, pro které definice chce uživatel spustit funkci.

Pokud uživatel funkci spouštěl nad seznamem, zobrazil jsem dotaz, zda chce funkci opravdu spustit. Následovalo načtení ID označených záznamů a ty jsem si uložil do dočasné tabulky. Pokud byly označené i neaktivní definice, tak byly ignorovány.

Při spuštění z příkazové řádky už žádný dotaz uživateli zobrazen nebyl. Následovalo načtení nepovinného parametru pořadí, pokud uživatel tento parametr nevyplnil uložil jsem do dočasné tabulky všechny aktivní definice. Pokud byl parametr vyplněn, tak jsem načetl pouze aktivní definice, které mělo pořadí, zapsané v parametru.

Když jsem měl v dočasné tabulce všechny ID definic, pro které se měla funkce spustit vyvolal jsem databázovou proceduru. V proceduře jsem si rozebral data z definic. Vytvořil jsem si další dočasné tabulky. První sloužila pro evidenci partnerů, pokud uživatel zadal jenom partnera byl zde uložen pouze jeden záznam. Při zadané skupině partnerů jsem si do tabulky vložil všechny partnery skupiny. Další dočasná tabulka sloužila k evidenci sortimentů. Do té jsem vkládal všechny sortimenty, které měly primárního dodavatele jednoho z partnerů v dočasné tabulce. Další dočasná tabulka sloužila pro evidenci skladů, zde jsem zvolil stejný postup jako u partnerů. Poslední dočasná tabulka sloužila pro samotné definice. Tuto tabulku jsem si musel zakládat pro rozlišení zadaných hodnot, případně zde uložit hodnoty již stávajících záznamů nebo záznamů nadřazených skupin. Postup pro vložení dat jsem zvolil takový, že pokud byla zadána hodnota

uložil jsem jí, pokud bylo zatrženo pole nezadáno, pokusil jsem se najít již existující záznam v parametrech výpočtu návrhu množství objednávky podle sortimentu a skladu, pokud jsem tento záznam nenašel, vzal jsem hodnotu z parametru pro skupinu sortimentu neboli nadřazeného záznamu. S dočasnou tabulkou a tabulkou parametrů výpočtu jsem provedl *MERGE* a tím jsem založil nebo aktualizoval záznamy.

Jako poslední krok jsem již pouze vypsal uživateli, pokud akci pouštěl nad seznamem, že akce proběhla v pořádku.

6 Závěr

Výsledkem mé práce ve společnosti je rozšíření produktu VENTUS®. Přesněji agendy sloužící pro návrh množství objednávky dodavateli. Kde jsem pomáhal při vývoji nového způsobu návrhu objednávání a také jsem vytvořil novou funkčnost pro hromadné nastavení parametrů pro výpočet návrhu.

Během absolvování stáže jsem velmi ocenil možnosti kontroly kódu zkušenějšími kolegy, kteří mi dokázali vysvětlit a navrhnout lepší řešení, než jsem původně zamýšlel já, ale také opravit chyby, kterým bych se měl v praxi vyvarovat. Při práci na úkolech jsem si také uvědomil, jak moc je důležitá dobrá dokumentace, hlavně pokud se vracím k úkolu, kterému jsem se věnoval před delší dobou, což v praxi nastává často.

Během praxe jsem mohl uplatnit své teoretické znalosti získané během studia. Nejvíce jsem využil znalostí databázových systému z předmětů *Úvod do databázových systémů* a *Databázové a informační systémy*. A také se znalostmi OOP z jiných jazyků, se kterými jsem pracoval během studia na vysoké škole nebylo vůbec složité naučit se nový jazyk Powercript.

Působení ve společnosti hodnotím velmi pozitivně a také jako důležitou část mého studia na vysoké škole. Věřím, že nově nabyté praktické zkušenosti mi pomohou při následném studiu, a také v budoucím hledání zaměstnání.

Literatura

[1] Apache™ Subversion®. [online], [cit 31. 3. 2019]. Dostupné z http://subversion.apache.org/.

[2] Introduction to CSS. [online], [cit 31. 3. 2019]. Dostupné z https://www.w3schools.com/css/css_intro.asp.

[3] Introduction to HTML. [online], [cit 31. 3. 2019]. Dostupné z https://www.w3schools.com/html/html_intro.asp.

[4] Kendo UI. [online], [cit 31. 3. 2019]. Dostupné z https://www.telerik.com/kendo-ui.

[5] Kvados a.s. - o společnosti. [online], [cit 31. 3. 2019]. Dostupné z https://www.kvados.cz/o-spolecnosti.

[6] SAP to Acquire Sybase, Inc. [online], [cit 31. 3. 2019]. Dostupné z https://www.sap.com/index.html?pressid=13202.

[7] Appeon Signs Agreement with SAP to Bring Major Innovations to PowerBuilder. [online], [cit 31. 3. 2019]. Dostupné z https://blogs.sap.com/2016/07/05/appeon-signs-agreement-with-sap-to-bring-major-innovations-to-powerbuilder-2/.

[8] SQL Introduction. [online], [cit 31. 3. 2019]. Dostupné z https://www.w3schools.com/sql/sql_intro.asp.

[9] SyBooks Online. [online], [cit 31. 3. 2019]. Dostupné z http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.dc00844.1250/html/pbug/CAIBIGBC.htm.

[10] TortoiseSVN. [online], [cit 31. 3. 2019]. Dostupné z https://tortoisesvn.net/.

[11] Transact-SQL. [online], [cit 31. 3. 2019]. Dostupné z https://searchsqlserver.techtarget.com/definition/T-SQL.

[12] VENTUS® - ERP systém. [online], [cit 31. 3. 2019]. Dostupné z https://www.kvados.cz/produkty/ventus.

[13] Introduction to XSLT. [online], [cit 2. 4. 2019]. Dostupné z https://www.w3schools.com/xml/xsl_languages.asp.